

A Parameterized Algorithmics Framework for Degree Sequence Completion Problems in Directed Graphs

Robert Brederick, Vincent Froese, Marcel Koseler, Marcelo Garlet Millani, André Nichterlein*, and Rolf Niedermeier

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany,
`{robert.bredereck,vincent.froese,andre.nichterlein,rolf.niedermeier}@tu-berlin.de`

Abstract

There has been intensive work on the parameterized complexity of the typically NP-hard task to edit undirected graphs into graphs fulfilling certain given vertex degree constraints. In this work, we lift the investigations to the case of directed graphs; herein, we focus on arc insertions. To this end, our general two-stage framework consists of efficiently solving a problem-specific number problem transferring its solution to a solution for the graph problem by applying flow computations. In this way, we obtain fixed-parameter tractability and polynomial kernelizability results, with the central parameter being the maximum vertex in- or outdegree of the output digraph. Although there are certain similarities with the much better studied undirected case, the flow computation used in the directed case seems not to work for the undirected case while f -factor computations as used in the undirected case seem not to work for the directed case.

1 Introduction

Modeling real-world networks (e.g., communication, ecological, social) often requests *directed* graphs (digraphs for short). We study a class of specific “network design” (in the sense of constructing a specific network topology) or “graph realization” problems. Here, our focus is on inserting arcs into a given digraph in order to fulfill certain vertex degree constraints. These problems are typically NP-hard, so we choose parameterized algorithm design for identifying relevant tractable special cases. The main parameter we work with is the maximum in- or outdegree of the newly constructed digraph. To motivate the problems we deal with, consider the following three application scenarios.

1. Assume we are given a directed network representing a system’s current state. Then, each individual node might have certain desired states of connectivity in terms of the numbers of in- and outgoing arcs which we want to satisfy by inserting arcs between the nodes. For instance, in a peer-review network we have an arc from one author reviewing a paper of another author. Depending on research experience, the authors might have different requests with respect to the number of own papers to be reviewed

*From February 2016 to January 2017 on postdoctoral leave to Durham University (GB), funded by DAAD.

by others and other papers which they are reviewing. This leads to the DIGRAPH DEGREE CONSTRAINT COMPLETION problem as studied in Section 4.1.

2. Assume that we have two different data sources: A network which is an incomplete measurement of some unreliable source and the true degree sequence of the target network. The goal is to reconstruct the original network by inserting arcs such that we obtain the target degree sequence (in a sense, the network matches the given degree sequence). In the presence of labeled input networks this might for example reveal communication patterns between users in social networks. The corresponding problem is called DIGRAPH DEGREE SEQUENCE COMPLETION and studied in Section 4.2.
3. Assume we want to “ k -anonymize” a social network, that is, after inserting a minimum number of arcs each degree, that is, each combination of in- and outdegree, occurs either zero or at least k times. This leads to the DIGRAPH DEGREE ANONYMITY problem as studied in Section 4.3.

All three problems are NP-hard. Based on a general framework presented in Section 3, we derive several fixed-parameter tractability results for them, mainly exploiting the parameter “maximum vertex degree” in the output digraph. Moreover, the three problems above are special cases of the DIGRAPH DEGREE CONSTRAINT SEQUENCE COMPLETION problem which we will define next. Before doing so, however, we want to go into a little more detail concerning the roots of the underlying graph-theoretic problems studied here. Since early computer science and algorithmic graph theory days, studies on the graph realizability of degree sequences (that is, multisets of positive integers or integer pairs) have played a prominent role, being performed both for undirected graphs [12, 21] as well as digraphs [6, 15, 22, 27]. Lately, the graph modification view gained more and more attention: given a graph, can it be changed by a minimum number of graph modifications such that the resulting graph adheres to specific constraints for its degree sequence?

In the most basic variant a degree sequence is a sequence of positive integers specifying (requested) vertex degrees for a fixed ordering of the vertices. Typically, the corresponding computational problems are NP-hard. In recent years, research in this direction focused on undirected graphs [14, 18, 19, 24, 32, 34]. In this work, we investigate parameterized algorithms on digraphs. As Gutin and Yeo [20] observed, much less is known about the structure of digraphs than that of undirected graphs making the design of parameterized algorithms for digraphs more challenging. In particular, we present a general framework for a class of degree sequence modification problems, focusing on the case of arc insertions (that is, completion problems).

The most general degree completion problem for digraphs we consider in this work is as follows.

DIGRAPH DEGREE CONSTRAINT SEQUENCE COMPLETION (DDConSeqC)

Input: A digraph $D = (V, A)$, a non-negative integer s , a “degree list function” $\tau: V \rightarrow 2^{\{0, \dots, r\}^2}$, and a “sequence property” Π .

Question: Is it possible to obtain a digraph D' by inserting at most s arcs in D such that the degree sequence of D' fulfills Π and $\deg_{D'}(v) \in \tau(v)$ for all $v \in V$?

We emphasize that there are two types of constraints—one (specified by the function τ) for the individual vertices and one (specified by Π) for the whole list of degree tuples. For instance, a common Π as occurring in the context of data privacy applications is to request

that the list is *k-anonymous*, that is, every combination of in- and outdegree that occurs in the list occurs at least k times (see the third motivating example before).

Since DDCONSEQC and its special cases as studied here all turn out to be NP-hard [28, 33], a parameterized complexity analysis seems the most natural fit for understanding the computational complexity landscape of these kinds of problems—this has also been observed in the above mentioned studies for the undirected case. Our main findings are mostly on the positive side. That is, although seemingly more intricate to deal with due to the existence of in- and outdegrees, many positive algorithmic results which hold for undirected graphs can also be achieved for digraphs (albeit using different techniques). In particular, we present a maximum-flow-based framework that, together with the identification and solution of certain number problems, helps to derive several fixed-parameter tractability results with respect to the parameter maximum possible in- or outdegree Δ^* in any solution digraph. Notably, the corresponding result in the undirected case was based on f -factor computations [14] which do not transfer to the directed case, and, vice versa, the flow computation approach we present for the directed case seemingly does not transfer to the undirected case. For special cases of DDCONSEQC, we can move further and even derive some polynomial-size problem kernels, again for the parameter Δ^* .

We consider the parameter Δ^* for the following reasons. First, it is always at most r , a natural parameter in the input. Second, in combination with Π , we might get an even smaller upper bound for Δ^* . Third, bounded-degree graphs are well studied and our work extends this since we only require Δ^* to be small, not to be constant. Fourth, in practice, the maximum degree is often significantly smaller than the number of vertices: Leskovec and Horvitz [30] studied a huge instant-messaging network (180 million vertices) with maximum degree 600. Furthermore, in the context of anonymization we expect that the maximum degree will not increase during the anonymization process [23]. Thus, the parameter Δ^* is interesting when studying kernelization as we do.

1.1 Related Work

Most of the work on graph modification problems for realizing degree constraints has focused on undirected graphs [14, 18, 19, 24, 32, 34]. Closest to our work is the framework for deriving polynomial-size problem kernels for undirected degree sequence completion problems [14], which we complement by our results for digraphs. Generally, we can derive similar results, but the technical details differ and the landscape of problems is richer in the directed case. As to digraph modification problems in general, we are aware of surprisingly little work. We mention work studying arc insertion for making a digraph transitive [36] or for making a graph Eulerian [10], both employing the toolbox of parameterized complexity analysis. Somewhat related is also work about the insertion of edges into a mixed graph to satisfy local edge-connectivity constraints [2] or about orienting edges in a partially oriented graph to make it an oriented graph [3].

1.2 Our Results

In Section 3, we present our general framework for DDCONSEQC. That is, based on flow computations, in a two-stage approach we show that it is fixed-parameter tractable with respect to the parameter Δ^* . To this end, we identify a specific pure number problem that needs to be fixed-parameter tractable with respect to the largest integer in the input. Next, present-

ing applications of the framework, in Section 4.1, we show that if there is no constraint Π concerning the degree sequence (that is, DIGRAPH DEGREE CONSTRAINT COMPLETION), then we not only obtain fixed-parameter tractability but also a polynomial-size problem kernel for parameter Δ^* can be obtained. Then, in Section 4.2 we show an analogous result if there is one exactly specified degree sequence to be realized (DIGRAPH DEGREE SEQUENCE COMPLETION). Finally, in Section 4.3, we show that if we request the degree sequence to be k -anonymous (that is, DIGRAPH DEGREE ANONYMITY), then we can at least derive a polynomial-size problem kernel for the combined parameter (s, Δ_D) , where Δ_D denotes the maximum in- or outdegree of the input digraph D . Also, we take a first step outlining the limitations of our framework for digraphs. In contrast to the undirected case (which is polynomial-time solvable [31]), the corresponding number problem of DIGRAPH DEGREE ANONYMITY surprisingly is weakly NP-hard and presumably not polynomial-time solvable.

2 Preliminaries

Notation. We consider *digraphs* (without multiarcs or loops) $D = (V, A)$ with $n := |V|$ and $m := |A|$. For a vertex $v \in V$, $\deg_D^-(v)$ denotes the *indegree* of v , that is, the number of incoming arcs of v . Correspondingly, $\deg_D^+(v)$ denotes the *outdegree*, that is, the number of outgoing arcs of v . We define the *degree* $\deg_D(v) := (\deg_D^-(v), \deg_D^+(v))$. The set $V(A') := \{v \in V \mid ((v, w) \in A' \vee (w, v) \in A') \wedge w \in V\}$ contains all vertices incident to an arc in $A' \subseteq V^2$. For a set of arcs $A' \subseteq V^2$, $D + A'$ denotes the digraph $(V, A \cup A')$, while $D[A']$ denotes the subdigraph $(V(A'), A')$. Analogously, for a set of vertices $V' \subseteq V$, $D[V']$ denotes the induced subdigraph $(V', A \cap (V')^2)$ which only contains the vertices V' and the arcs between vertices from V' . The set $N_D^+(v) := \{w \in V \mid (v, w) \in A\}$ denotes the set of *outneighbors* of v . Analogously, $N_D^-(v) := \{w \in V \mid (w, v) \in A\}$ denotes the set of *inneighbors*. Furthermore, we define the maximum indegree $\Delta_D^- := \max_{v \in V} \deg_D^-(v)$, the maximum outdegree $\Delta_D^+ := \max_{v \in V} \deg_D^+(v)$, and $\Delta_D := \max\{\Delta_D^+, \Delta_D^-\}$.

A *digraph degree sequence* $\sigma = \{(d_1^-, d_1^+), \dots, (d_n^-, d_n^+)\}$ is a multiset of nonnegative integer tuples, where $d_i^-, d_i^+ \in \{0, \dots, n-1\}$ for all $i \in \{1, \dots, n\}$. We define

$$\begin{aligned} \Delta_\sigma^- &:= \max\{d_1^-, \dots, d_n^-\}, \\ \Delta_\sigma^+ &:= \max\{d_1^+, \dots, d_n^+\}, \text{ and} \\ \Delta_\sigma &:= \max\{\Delta_\sigma^-, \Delta_\sigma^+\}. \end{aligned}$$

For a digraph $D = (\{v_1, \dots, v_n\}, A)$ we denote by $\sigma(D) := \{\deg_D(v_1), \dots, \deg_D(v_n)\}$, the digraph degree sequence of D . Let $d = (d^-, d^+)$ be a nonnegative integer tuple. For a digraph D , the *block* $B_D(d)$ of *degree* d is the set of all vertices having degree d , formally $B_D(d) := \{v \in V \mid \deg_D(v) = d\}$. We define $\lambda_D(d)$ as the number of vertices in D with degree d , that is, $\lambda_D(d) := |B_D(d)|$. Similarly, we define $B_\sigma(t)$ as the multiset of all tuples equal to t and $\lambda_\sigma(t)$ as the number of occurrences of the tuple t in the multiset σ . For two integer tuples $(x_1, y_1), (x_2, y_2)$, we define the sum $(x_1, y_1) + (x_2, y_2) := (x_1 + x_2, y_1 + y_2)$.

Parameterized Algorithmics. We assume the reader to be familiar with classical complexity theory concepts such as polynomial-time reductions and (weak) NP-hardness [1, 17]. An instance (I, k) of a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ consists of the classical input I and a *parameter* k . A parameterized problem L is called *fixed-parameter tractable* (fpt) with

respect to the parameter k if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is a function only depending on k and $|I|$ denotes the size of the input I . Accordingly, for a *combined parameter* (k_1, k_2, \dots) , a parameterized problem is fpt if it can be solved in $f(k_1, k_2, \dots) \cdot |I|^{O(1)}$ time.

A *kernelization* is a polynomial-time algorithm transforming a given instance (I, k) into an equivalent instance (I', k') with $|I'| \leq g(|I|)$ and $k' \leq h(k)$ for some functions g and h , that is, (I, k) is a yes-instance if and only if (I', k') is a yes-instance. The instance (I', k') is called the *problem kernel* and g denotes its size. If g is a polynomial, then we have a *polynomial(-size)* problem kernel. It can be shown that a parameterized problem is fpt if and only if it has a problem kernel. For further details, we refer the reader to the books by Downey and Fellows [11] and Cygan et al. [9].

3 The Framework

Our goal is to develop a framework for deriving fixed-parameter tractability for a general class of completion problems in directed graphs. To this end, recall our general setting for DD-CONSEQC which is as follows. We are given a digraph and want to insert at most s arcs such that the vertices satisfy certain degree constraints τ , and additionally, the degree sequence of the digraph fulfills a certain property Π . Formally, the sequence property Π is given as a function that maps a digraph degree sequence to 1 if the sequence fulfills the property and otherwise to 0. We restrict ourselves to properties where the corresponding function can be encoded with only polynomially many bits in the number of vertices of the input digraph and can be decided efficiently.¹ We remark that it is not always the case that there are both vertex degree constraints (as defined by τ) and degree sequence constraints (as defined by Π) requested. This can be handled by either setting τ to the trivial degree list function with $\tau(v) = \{0, \dots, n-1\}$ ² for all $v \in V$ or setting Π to allow all possible degree sequences.

In this section, we show how to derive (under certain conditions) fixed-parameter tractability with respect to the maximum possible in- or outdegree Δ^* of the *output* digraph for DD-CONSEQC. Note that Δ^* in general is not known in advance. In practice, we might therefore instead consider upper bounds for Δ^* which depend on the given input. For example, it always holds $\Delta^* \leq \min\{r, \Delta_D + s\}$ since we are only inserting at most s arcs in D . Clearly, Δ^* might also be upper-bounded depending on Π (or even depending on r , s , Δ_D , and Π) in some cases. Our generic framework consists of two main steps: First, we prove fixed-parameter tractability with respect to the combined parameter (s, Δ_D) in Section 3.1. This step generalizes ideas for the undirected case [14]. Note that $\Delta_D \leq \Delta^*$ trivially holds. Second, we show in Section 3.2 how to upper-bound the number s of arc insertions polynomially in Δ^* by solving a certain problem specific numerical problem. For this step, we develop a new key argument based on a maximum flow computation (the undirected case was based on f -factor arguments).

¹All specific properties in this work can be easily decided in polynomial time. Indeed, in many cases even fixed-parameter tractability with respect to the maximum integer in the sequence would suffice.

3.1 Fixed-parameter tractability with respect to (s, Δ_D)

We show that DDCONSEQC is fixed-parameter tractable with respect to the combination of the maximum number s of arcs to insert and the maximum in- or outdegree Δ_D of the input digraph D . The basic idea underlying this result is that two vertices v and w with $\deg_D(v) = \deg_D(w)$ and $\tau(v) = \tau(w)$ are interchangeable. Accordingly, we will show that it suffices to consider only a bounded number of vertices with the same “degree properties”. In particular, if there is a solution, then there is also a solution that only inserts arcs between a properly chosen subset of vertices of bounded size. To formalize this idea, we introduce the notion of an α -block-type set for some positive integer α .

To start with, we define the types of a vertex via the numbers of arcs that τ allows to add to this vertex. Let (D, s, τ, Π) be a DDCONSEQC instance. A vertex v is of *type* $t \in \{0, \dots, \Delta^*\}^2$ if $\deg_D(v) + t \in \tau(v)$. Observe that one vertex can be of several types. The subset of $V(D)$ containing all vertices of type t is denoted by $T_{D,\tau}(t)$. A vertex v of type $(0, 0)$ (that is, $\deg_D(v) \in \tau(v)$) is called *satisfied*. A vertex which is not satisfied is called *unsatisfied*. We next define our notion of α -block-type sets and its variants.

Definition 1. Let α be a positive integer and let $U \subseteq V(D)$ denote the set of all unsatisfied vertices in D . A vertex subset $C \subseteq V(D)$ with $U \subseteq C$ is called

- α -type set if, for each type $t \neq (0, 0)$, C contains exactly $\min\{|T_{D,\tau}(t) \setminus U|, \alpha\}$ satisfied vertices of type t ;
- α -block set if, for each degree $d \in \sigma(D)$, C contains exactly $\min\{|B_D(d) \setminus U|, \alpha\}$ satisfied vertices with degree d ;
- α -block-type set if, for each degree $d \in \sigma(D)$ and each type $t \neq (0, 0)$, C contains exactly $\min\{|(B_D(d) \cap T_{D,\tau}(t)) \setminus U|, \alpha\}$ satisfied vertices of degree d and type t .

As a first step, we prove that these sets defined above can be computed efficiently.

Lemma 2. An α -type/ α -block/ α -block-type set C as described in Definition 1 can be computed in $O(m + |\tau| + r^2)$ / $O(m + n + \Delta_D^2)$ / $O(m + |\tau| + \Delta_D^2 r^2)$ time.

Proof. To compute an α -block-type set, we start with an empty set $C := \emptyset$ and for each possible vertex degree d and each possible vertex type t , we initialize a counter $x(d, t) := 0$. We then iterate over all vertices $v \in V(D)$. If v is unsatisfied, that is $\deg_D(v) \notin \tau(v)$, then we add v to C . If v is satisfied, then for each type $t \neq (0, 0)$ with $\deg_D(v) + t \in \tau(v)$, we increase the counter $x(\deg_D(v), t)$ by one and add v to C if $x(\deg_D(v), t) < \alpha$. This can be done in $O(m + |\tau| + \Delta_D^2 r^2)$ time. The other two cases of computing an α -block set or an α -type set can be done in a similar fashion. \square \square

We move on to the crucial lemma stating that a solution (that is, a set of arcs), if existing, can always be found in between vertices of an α -block-type set C given that C contains “enough” vertices of each degree and type. Here, enough means $\alpha := 2s(\Delta_D + 1)$.

Lemma 3. Let (D, s, τ, Π) be a DDCONSEQC instance and let $C \subseteq V(D)$ be a $2s(\Delta_D + 1)$ -block-type set. If (D, s, τ, Π) is a yes-instance, then there exists a solution $A^* \subseteq C^2$ for (D, s, τ, Π) , that is, $|A^*| \leq s$, $\sigma(D + A^*)$ fulfills Π , and $\deg_{D+A^*}(v) \in \tau(v)$ for all $v \in V(D)$.

Proof. Let $A' \subseteq V(D)^2 \setminus A(D)$ be a solution for (D, s, τ, Π) that minimizes the number of vertices not in C , that is, $|V(A') \setminus C|$ is minimum. The solution A' exists since (D, s, τ, Π) is a yes-instance. If $V(A') \subseteq C$, then we are done. Hence, we assume that there exists a vertex v in $V(D) \setminus C$ which is incident to at least one arc in A' . Let $V_v^- := \{u \mid (u, v) \in A'\}$ and let $V_v^+ := \{w \mid (v, w) \in A'\}$ be the set of in- respectively outneighbors of v in A' . Furthermore, let $d := \deg_D(v)$ and $t := (|V_v^-|, |V_v^+|)$. Thus, v has degree d and is of type t . By definition of C , it follows that $|B_D(d) \cap T_{D,\tau}(t)| > 2s(\Delta_D + 1)$.

Now, we claim that there is a vertex $v^* \in (B_D(d) \cap T_{D,\tau}(t) \cap C) \setminus V(A')$ such that we can replace v with v^* in the solution. More precisely, in all arcs of A' we want to replace v by v^* , that is, we obtain a new arc set $A^* := \{(u, w) \in A' \mid u \neq v \wedge w \neq v\} \cup \{(u, v^*) \mid u \in V_v^-\} \cup \{(v^*, w) \mid w \in V_v^+\}$. Since we cannot insert arcs that already exist in the input digraph D , we need that $N_D^-(v^*) \cap V_v^- = \emptyset$ and $N_D^+(v^*) \cap V_v^+ = \emptyset$. Observe that such a vertex v^* exists: Since each of the at most s vertices in $V_v^+ \cup V_v^-$ has at most Δ_D incoming and Δ_D outgoing arcs, it follows that at most $s \cdot 2\Delta_D$ vertices in $B_D(d) \cap T_{D,\tau}(t) \cap C$ can have an arc from or to a vertex in $V_v^+ \cup V_v^-$. Furthermore, since $|A'| \leq s$, it follows that at most $2s - 1$ vertices in $B_D(d) \cap T_{D,\tau}(t) \cap C$ are incident to an arc in A' (the minus one comes from the fact that v is incident to at least one arc in A'). By definition of C , it follows that $|B_D(d) \cap T_{D,\tau}(t) \cap C| \geq 2s(\Delta_D + 1) > s \cdot 2\Delta_D + 2s - 1$. Hence, there is at least one vertex $v^* \in B_D(d) \cap T_{D,\tau}(t) \cap C$ that is not adjacent to any vertex in $V_v^+ \cup V_v^-$ and not incident to any arc in A' . Thus, we can replace v by v^* .

We now show that A^* is still a solution: First, observe that $\sigma(D + A') = \sigma(D + A^*)$ and, hence, $\sigma(D + A^*)$ fulfills Π . Second, observe that $\deg_{D+A^*}(v) \in \tau(v)$ since $v \notin C$, which implies that v was satisfied. Furthermore, $\deg_{D+A^*}(v^*) \in \tau(v^*)$ since v^* is of type t . Hence, A^* is a solution and $|V(A') \setminus C| > |V(A^*) \setminus C|$, a contradiction to the assumption that A' was a solution minimizing this value. \square \square

If there are no restrictions on the resulting degree sequence (as it is the case for the DIGRAPH DEGREE CONSTRAINT COMPLETION problem (DDCONC) in Section 4.1), then we can replace the $2s(\Delta_D + 1)$ -block-type set in Lemma 3 by a $2s(\Delta_D + 1)$ -type set:

Lemma 4. *Let (D, s, τ) be a DDCONC instance and let $C \subseteq V(D)$ be a $2s(\Delta_D + 1)$ -type set. If (D, s, τ) is a yes-instance, then there exists a solution $A^* \subseteq C^2$ for (D, s, τ) , that is, $|A^*| \leq s$ and $\deg_{D+A^*}(v) \in \tau(v)$ for all $v \in V(D)$.*

Similarly, if there are no restrictions on the individual vertex degrees, that is, τ is the degree list function $\tau(v) = \{0, \dots, n-1\}^2$ for all $v \in V(D)$, then we can replace the $2s(\Delta_D + 1)$ -block-type set by a $2s(\Delta_D + 1)$ -block set.

Lemma 5. *Let (D, s, τ, Π) be a DDONSEQC instance where $\tau(v) = \{0, \dots, n-1\}^2$ for all $v \in V(D)$ and let $C \subseteq V(D)$ be a $2s(\Delta_D + 1)$ -block set. If (D, s, τ, Π) is a yes-instance, then there exists a solution $A^* \subseteq C^2$ for (D, s, τ, Π) , that is, $|A^*| \leq s$ and $\sigma(D + A^*)$ fulfills Π .*

Lemma 3 implies a fixed-parameter algorithm by providing a bounded search space for possible solutions, namely any $2s(\Delta_D + 1)$ -block-type set C .

Theorem 6. *If deciding Π is fixed-parameter tractable with respect to the maximum integer in the input sequence, then DDONSEQC is fixed-parameter tractable with respect to (s, Δ_D) .*

Proof. Given a DDONSEQC instance (D, s, τ, Π) , we first check in polynomial time whether there are more than $2s$ unsatisfied vertices in D . If this is the case, then we have a no-instance,

since we can change the degrees of at most $2s$ vertices by inserting at most s arcs. Otherwise, we compute a $2s(\Delta_D + 1)$ -block-type set C in polynomial time (Lemma 2). By Lemma 3, we know that it is sufficient to search for a solution within the vertices of C . Hence, we simply try out all possible arc sets $A' \subseteq C^2$ of size at most s and check whether in one of the cases the vertex degrees and the degree sequence of $D + A'$ satisfy the requirements τ and Π . Since C contains at most $2s$ unsatisfied vertices and at most $2s(\Delta_D + 1) \cdot (\Delta_D + 1)^2 (\Delta^*)^2$ satisfied vertices, and since $\Delta^* \leq \Delta_D + s$, there are at most $O(2^{(2s+2s(\Delta_D+1)^3(\Delta_D+s)^2)^2})$ possible subsets of arcs to insert. Checking whether τ is satisfied can be done in polynomial time and deciding whether Π holds for $\sigma(D + A')$ is by assumption fixed-parameter tractable with respect to the largest integer, which is at most $\Delta_D + s$. Thus, overall, we obtain fixed-parameter tractability with respect to (s, Δ_D) . \square \square

3.2 Bounding the solution size s polynomially in Δ^*

This subsection constitutes the major part of our framework. The rough overall scheme is analogous to the undirected case as described by Froese et al. [14]. By dropping the graph structure and solving a simpler problem-specific number problem on the degree sequence of the input digraph, we show how to solve DDCONSEQC instances with “large” solutions provided that we can solve the associated number problem efficiently. The number problem is defined so as to simulate the insertion of arcs to a digraph on an integer tuple sequence. Note that inserting an arc increases the indegree of a vertex by one and increases the outdegree of another vertex by one. Inserting s arcs can thus be represented by increasing the tuple entries in the degree sequence by an overall value of s in each component. Formally, the corresponding number problem (abbreviated as #DDCONSEQC) is defined as follows.

NUMBERS ONLY DIGRAPH DEGREE CONSTRAINT SEQUENCE COMPLETION

Input: A sequence $\sigma = (c_1, d_1), \dots, (c_n, d_n)$ of n nonnegative integer tuples, a positive integer s , a “tuple list function” $\tau: \{1, \dots, n\} \rightarrow 2^{\{0, \dots, r\}^2}$, and a sequence property Π .

Question: Is there a sequence $\sigma' = (c'_1, d'_1), \dots, (c'_n, d'_n)$ such that $\sum_{i=1}^n c'_i - c_i = \sum_{i=1}^n d'_i - d_i = s$, $c_i \leq c'_i$, $d_i \leq d'_i$, and $(c'_i, d'_i) \in \tau(i)$ for all $1 \leq i \leq n$, and σ' fulfills Π ?

If we plug the degree sequence of a digraph into #DDCONSEQC, then an integer tuple (c'_i, d'_i) of a solution tells us to add $x_i := c'_i - c_i$ incoming arcs and $y_i := d'_i - d_i$ outgoing arcs to the vertex v_i . We call the tuples (x_i, y_i) *demands*. Having computed the demands, we can then try to solve our original DDCONSEQC instance by searching for a set of arcs to insert that exactly fulfills the demands. Such an arc set, however, might not always exist. Hence, the remaining problem is to decide whether it is possible to realize the demands in the given digraph. The following lemma shows (using flow computations) that this is in fact always possible if the number s of arcs to insert is large compared to Δ^* .

Lemma 7. *Let $D = (V = \{v_1, \dots, v_n\}, A)$ be a digraph and let x_1, \dots, x_n , y_1, \dots, y_n , and Δ^* be nonnegative integers such that*

- (I) $\Delta^* \leq n - 1$,
- (II) $\deg_D^-(v_i) + x_i \leq \Delta^*$ for all $i \in \{1, \dots, n\}$,
- (III) $\deg_D^+(v_i) + y_i \leq \Delta^*$ for all $i \in \{1, \dots, n\}$,
- (IV) $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i =: s$, and
- (V) $s > 2(\Delta^*)^2$.

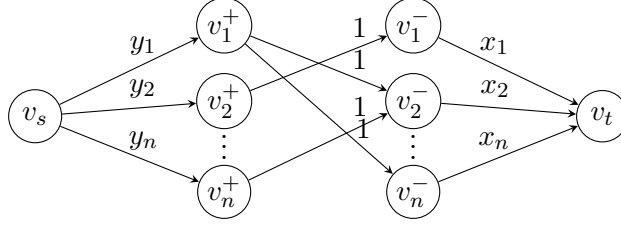


Figure 1: A flow network as described in Construction 8. For each vertex v_i in the digraph D there are two vertices v_i^+ and v_i^- . We connect a vertex v_i^+ to a vertex v_j^- if the arc (v_i, v_j) is not in D . Inserting the arc (v_i, v_j) is then represented by setting the flow on the arc (v_i^+, v_j^-) to one.

Then, there exists an arc set $A' \subseteq V^2 \setminus A$ of size s such that for the digraph $D' := D + A'$ it holds $\deg_{D'}(v_i) = \deg_D(v_i) + (x_i, y_i)$ for all $v_i \in V$. Moreover, the set A' can be computed in $O(n^3)$ time.

Proof. The proof is based on a flow network which we construct such that the corresponding maximum flow yields the set A' of arcs to be inserted in D in order to obtain our target digraph D' .

Construction 8. We build a flow network $N = (V_N, A_N)$ according to the following steps.

- Add a source vertex v_s and a sink vertex v_t to N ;
- for each vertex $v_i \in V$, add two vertices v_i^+, v_i^- to N ;
- for each $i \in \{1, \dots, n\}$, insert the arc (v_s, v_i^+) with capacity y_i ;
- for each $i \in \{1, \dots, n\}$, insert the arc (v_i^-, v_t) with capacity x_i ;
- for each $(v_i, v_j) \in V^2 \setminus A$ with $i \neq j$, insert the arc (v_i^+, v_j^-) with capacity one.

The network N contains $|V_N| \in O(n)$ vertices and $|A_N| \in O(n^2 - m)$ arcs (since $m \leq n^2 - n$, we also have $|A_N| \in \Omega(n)$) and can be constructed in $O(n^2)$ time. See Figure 1 for an illustration. Inserting an arc (v_i, v_j) in D corresponds to sending flow from v_i^+ to v_j^- . Since, by definition, each vertex v_i^+ will only receive at most y_i flow from v_s and each vertex v_i^- will send at most x_i flow to v_t , we cannot insert more than s arcs (Condition (IV)).

We claim that for $s > 2(\Delta^*)^2$ (Condition (V)), the maximum flow in the network is indeed s . To see this, let $V_N^+ := \{v_i^+ \in V_N \mid i \in \{1, \dots, n\}\}$ and let $V_N^- := \{v_i^- \in V_N \mid i \in \{1, \dots, n\}\}$. In the following, a vertex $v_i^+ \in V_N^+$ ($v_j^- \in V_N^-$) is called *saturated* with respect to a flow $f: A_N \rightarrow \mathbb{R}^+$, if $f(v_s, v_i^+) = y_i$ ($f(v_j^-, v_t) = x_j$). Suppose that the maximum flow f has a value less than s . Then, there exist non-saturated vertices $v_i^+ \in V_N^+$ and $v_j^- \in V_N^-$. Let $X \subseteq V_N^-$ be the vertices to which v_i^+ has an outgoing arc in the residual graph and let $Y \subseteq V_N^+$ be the vertices which have an outgoing arc to v_j^- in the residual graph. Observe that $\deg_N^+(v_i^+) = n - 1 - \deg_D^+(v_i)$ and $\deg_N^-(v_j^-) = n - 1 - \deg_D^-(v_j)$. Consequently, $|X| \geq n - 1 - \deg_D^+(v_i) - y_i \geq n - 1 - \Delta^*$ holds due to Condition (III). Since v_i^+ is not saturated, we know that $|X| \geq n - \Delta^* \geq 1$ (due to Condition (I)). By the same reasoning (using Conditions (I) and (II)) it follows that $|Y| \geq n - \Delta^* \geq 1$.

Remember that f is a flow of maximum value. Hence, each vertex in X and each vertex in Y is saturated. Otherwise, there would be an augmenting path in the residual graph, contradicting our assumption of f being maximal. If a vertex $x \in X$ would receive flow from a vertex $y \in Y$, then this implies a backward arc in the residual graph resulting in an augmenting path $v_s \rightarrow v_i^+ \rightarrow x \rightarrow y \rightarrow v_j^- \rightarrow v_t$, again contradicting our maximality assumption for f . Thus, we can conclude that all the flow that goes into X has to come from the remaining vertices in $V_N^+ \setminus (Y \cup \{v_i^+\})$. This set has size at most $n - |Y| \leq n - (n - \Delta^*) = \Delta^*$. But since $y_\ell \leq \Delta^*$ for all $\ell \in \{1, \dots, n\}$ (by Condition (III)), those Δ^* vertices can cover at most a flow of value $(\Delta^*)^2$ and hence,

$$\sum_{v_i^- \in X} x_i \leq \sum_{v_i^+ \in V_N^+ \setminus (Y \cup \{v_i^+\})} y_i \leq (\Delta^*)^2. \quad (1)$$

Since X is saturated, and since also $x_\ell \leq \Delta^*$ holds for all $\ell \in \{1, \dots, n\}$ (Condition (II)), we obtain from Condition (IV)

$$\begin{aligned} s = \sum_{i=1}^n x_i &= \sum_{v_i^- \in X} x_i + \sum_{v_i^- \in V_N^- \setminus X} x_i \stackrel{(1)}{\leq} (\Delta^*)^2 + \sum_{v_i^- \in V_N^- \setminus X} \Delta^* \\ &= (\Delta^*)^2 + |V_N^- \setminus X| \cdot \Delta^* = (\Delta^*)^2 + n - |X| \cdot \Delta^* \\ &\leq (\Delta^*)^2 + \Delta^* \cdot \Delta^*. \end{aligned}$$

This contradicts $s > 2(\Delta^*)^2$ (Condition (V)) and hence proves the claim.

Now, let f be a maximum flow in N (computable in $O(|V_N||E_N|) = O(n(n^2 - m))$ time [35]) and let $A' := \{(v_i, v_j) \in V^2 \mid f((v_i^+, v_j^-)) = 1\}$ and note that $|A'| = s$ and $A' \cap A = \emptyset$. Clearly, for the digraph $D' := D + A'$ it holds $\deg_{D'}(v_i) = \deg_D(v_i) + (x_i, y_i)$ for all $v_i \in V$. \square \square

We remark that similar flow-constructions as given in the proof above have been used before [8, 16]. The difference here is that we actually argue about the size of the flow and not only about polynomial-time solvability. Consequently, our proof uses different arguments.

With Lemma 7 we have the key which allows us to transfer solutions of $\#DDCONSEQC$ to solutions of $DDCONSEQC$. The following lemma is immediate.

Lemma 9. *Let $I := (D = (V, A), s, \tau, \Pi)$ with $V = \{v_1, \dots, v_n\}$ be an instance of $DDCONSEQC$ with $s > 2(\Delta^*)^2$. If there exists an s' with $2(\Delta^*)^2 < s' \leq s$ such that $I' := (\deg_D(v_1), \dots, \deg_D(v_n), s', \tau', \Pi)$ with $\tau'(i) := \tau(v_i)$ for all $v_i \in V$ is a yes-instance of $\#DDCONSEQC$, then also I is a yes-instance of $DDCONSEQC$.*

We now have all ingredients for our first main result, namely transferring fixed-parameter tractability with respect to the combined parameter (s, Δ^*) to fixed-parameter tractability with respect to the single parameter Δ^* , provided that $\#DDCONSEQC$ is fixed-parameter tractable with respect to the largest possible integer ξ in the output sequence. The idea is to search for large solutions based on Lemma 9 using $\#DDCONSEQC$. If there are no large solutions (that is, $s \leq 2(\Delta^*)^2$), then we run an FPT-algorithm with respect to (s, Δ^*) .

Theorem 10. *If $DDCONSEQC$ is fixed-parameter tractable with respect to (s, Δ^*) and $\#DDCONSEQC$ is fixed-parameter tractable with respect to the largest possible integer ξ in the output sequence, then $DDCONSEQC$ is fixed-parameter tractable with respect to Δ^* .*

Proof. In the following, let \mathcal{A} be the fixed-parameter algorithm solving DDConSEQC in $h(s, \Delta^*) \cdot n^{O(1)}$ time and let \mathcal{A}' be the fixed-parameter algorithm solving #DDConSEQC in $h'(\xi) \cdot n^{O(1)}$ time. Let $I := (D = (V, A), s, \tau, \Pi)$ be a DDConSEQC instance.

If $s \leq 2(\Delta^*)^2$, then we can run algorithm \mathcal{A} on I in $h(s, \Delta^*) \cdot n^{O(1)} \leq g(\Delta^*) \cdot n^{O(1)}$ time for some function g .

Otherwise, we check for each $s' \in \{2(\Delta^*)^2 + 1, \dots, s\}$ whether the instance $I_{s'} := (\deg_D(v_1), \dots, \deg_D(v_n), s', \tau', \Pi)$ with $\tau'(i) := \tau(v_i)$ for all $v_i \in V$ is a yes-instance of #DDConSEQC using algorithm \mathcal{A}' . Note that the running time is at most $s \cdot h'(\Delta^*) \cdot n^{O(1)}$. If we find a yes-instance $I_{s'}$ for some s' , then we know by Lemma 9 that I is also a yes-instance. If $I_{s'}$ is a no-instance for all $s' \in \{2(\Delta^*)^2 + 1, \dots, s\}$, then we also know that there cannot exist a solution for I of size larger than $2(\Delta^*)^2$ since the existence of a solution for a DDConSEQC instance clearly implies a solution for the corresponding #DDConSEQC instance. Therefore, I is a yes-instance if and only if $I' := (D, 2(\Delta^*)^2, \tau, \Pi)$ is a yes-instance. We can thus run algorithm \mathcal{A} on I' in $h(2(\Delta^*)^2, \Delta^*) \cdot n^{O(1)}$ time. \square \square

Our second main result allows to transfer a polynomial-size problem kernel with respect to (s, Δ^*) to a polynomial-size problem kernel with respect to Δ^* if #DDConSEQC is polynomial-time solvable. The proof is analogous to the proof of Theorem 10.

Theorem 11. *If DDConSEQC admits a problem kernel containing $g(s, \Delta^*)$ vertices computable in $p(n)$ time and #DDConSEQC is solvable in $q(n)$ time for polynomials p and q , then DDConSEQC admits a problem kernel with $g(2(\Delta^*)^2, \Delta^*)$ vertices computable in $O(s \cdot q(n) + p(n))$ time.*

Proof. Let $I := (D = (V, A), s, \tau, \Pi)$ be a DDConSEQC instance. If $s \leq 2(\Delta^*)^2$, then we simply run the kernelization algorithm on I obtaining an equivalent instance containing at most $g(2(\Delta^*)^2, \Delta^*)$ vertices in $p(n)$ time. Otherwise, we check in $q(n)$ time for each $s' \in \{2(\Delta^*)^2 + 1, \dots, s\}$, whether the instance $I_{s'} := (\deg_D(v_1), \dots, \deg_D(v_n), s', \tau', \Pi)$ with $\tau'(i) := \tau(v_i)$ for all $v_i \in V$ is a yes-instance of #DDConSEQC. Note that the running time is thus at most $s \cdot q(n)$. If we find a yes-instance $I_{s'}$ for some s' , then we know by Lemma 9 that also I is a yes-instance, and thus we return a trivial DDConSEQC yes-instance. If $I_{s'}$ is a no-instance for all $s' \in \{2(\Delta^*)^2 + 1, \dots, s\}$, then we also know that there cannot exist a solution for I of size larger than $2(\Delta^*)^2$ since the existence of a solution for a DDConSEQC instance clearly implies a solution for the corresponding #DDConSEQC instance. Therefore, I is a yes-instance if and only if $I' := (D, 2(\Delta^*)^2, \tau, \Pi)$ is a yes-instance. Again, we run the kernelization algorithm on I' and return an equivalent instance with at most $g(2(\Delta^*)^2, \Delta^*)$ vertices in $p(n)$ time. The overall running time is thus in $O(s \cdot q(n) + p(n))$ and we obtain a problem kernel with respect to Δ^* . \square \square

4 Applications

In the following, we show how the framework described in Section 3 can be applied to three special cases of DDConSEQC. These special cases naturally extend known problems on undirected graphs to the digraph setting.



Figure 2: Two example instances of DDConc with $s = 1$. The left instance is solvable by inserting the (dashed) arc from the right vertex to the middle vertex. The right instance is a no-instance since one cannot add an outgoing arc to the left vertex or to the middle vertex but one has to add an incoming arc to the right vertex (loops are not allowed).

4.1 Digraph Degree Constraint Completion

In this section, we investigate the NP-hard special case of DDConc² where the property Π allows any possible degree sequence, see Figure 2 for two illustrating examples.

DIGRAPH DEGREE CONSTRAINT COMPLETION (DDConc)

Input: A digraph $D = (V, A)$, a positive integer s , and a “degree list function” $\tau: V \rightarrow 2^{\{0, \dots, r\}^2}$.

Question: Is it possible to obtain a digraph D' by inserting at most s arcs in D such that $\deg_{D'}(v) \in \tau(v)$ for all $v \in V$?

DDConc is the directed (completion) version of the well-studied undirected DEGREE CONSTRAINT EDITING problem [18, 32] for which an $O(r^5)$ -vertex problem kernel is known [14]. We subsequently transfer the polynomial-size problem kernel for the undirected case to a polynomial-size problem kernel for DDConc with respect to Δ^* . Note that the parameter Δ^* is clearly at most r . Since it is trivial to decide Π in this case, we obtain fixed-parameter tractability of DDConc with respect to (s, Δ_D) due to Theorem 6, which is based on a bounded search space, namely a $2s(\Delta_D + 1)$ -type set (see Definition 1 and Lemma 4). We further strengthen this result by removing all vertices that are not in the $2s(\Delta_D + 1)$ -type set and adjusting the degree list function τ appropriately. Lemma 4 then yields the correctness of this approach resulting in a polynomial-size problem kernel with respect to (s, Δ^*) .

We start with the following simple reduction rule. Recall that a vertex v is called unsatisfied if $\deg_D(v) \notin \tau(v)$.

Reduction Rule 4.1. *Let $(D = (V, A), s, \tau)$ be a DDConc instance. If there are more than $2s$ unsatisfied vertices, then return a trivial no-instance. Moreover, if there exists a vertex $v \in V$ with $\deg_D^-(v) > \Delta^*$ or $\deg_D^+(v) > \Delta^*$, then also return a trivial no-instance.*

Lemma 12. *Reduction Rule 4.1 is correct and can be computed in $O(m + |\tau|)$ time.*

Proof. If there are more than $2s$ unsatisfied vertices, then we can return a trivial no-instance since inserting an arc can satisfy at most two vertices. Also, by inserting arcs we can only increase in- and outdegrees of vertices. Hence, we can return a no-instance if the in- or outdegree of a vertex is larger than Δ^* . This proves the correctness.

The reduction rule is applicable in $O(m + |\tau|)$ time by computing the degree of each vertex in $O(n + m)$ time and subsequently iterating through τ . \square \square

Based on Reduction Rule 4.1, we obtain a polynomial-size problem kernel with respect to the combined parameter (s, Δ^*) as follows.

²This special case was investigated more specifically in the Bachelor thesis of Koseler [28] (online available).

Theorem 13. *DDCONC admits a problem kernel containing $O(s(\Delta^*)^3) \subseteq O(sr^3)$ vertices. It is computable in $O(m + |\tau| + r^2)$ time.*

Proof. Let $I = (D = (V, A), s, \tau)$ be an instance of DDCONC. First, we apply Reduction Rule 4.1 in $O(m + |\tau|)$ time. If a no-instance is returned, then we are done. Otherwise, we know that there are at most $2s$ unsatisfied vertices. Also, we know that $\deg_D^-(v) \leq \Delta^*$ and $\deg_D^+(v) \leq \Delta^*$ for all $v \in V$. We compute an α -type set C (see Definition 1) for $\alpha := 2s(\Delta_D + 1)$ in $O(m + |\tau| + r^2)$ time (Lemma 2) and return the instance $I' = (D' := D[C], s, \tau_C)$, where the adjusted degree list $\tau_C(v)$, for each $v \in C$, is defined as follows:

$$\tau_C(v) := \{(i, j) \in \{0, \dots, \Delta^*\}^2 \mid (i, j) + (|N_D^-(v) \setminus C|, |N_D^+(v) \setminus C|) \in \tau(v)\}.$$

The instance I' can be computed in $O(m + |\tau| + r^2)$ time. We now show that I' is an equivalent instance of DDCONC.

Assume that I' is a yes-instance, that is, there exists a set $A' \subseteq C^2$ of size at most s such that $\deg_{D'+A'}(v) \in \tau_C(v)$ for each $v \in C$. Then, the set A' is also a solution for I since for each vertex $v \in C$, it holds

$$\deg_{D+A'}(v) = \deg_{D'+A'}(v) + (|N_D^-(v) \setminus C|, |N_D^+(v) \setminus C|) \in \tau(v),$$

by definition of $\tau_C(v)$. Moreover, for each vertex $v \in V \setminus C$, we have $\deg_{D+A'}(v) = \deg_D(v) \in \tau(v)$ since $V \setminus C$ contains only satisfied vertices. Hence, I is a yes-instance.

Conversely, let I be a yes-instance. Then, by Lemma 4, we know that there exists an arc set $A^* \subseteq C^2$ of size at most s such that $\deg_{D+A^*}(v) \in \tau(v)$ for all $v \in V$. Then, for each vertex $v \in C$, it holds

$$\deg_{D'+A^*}(v) = \deg_{D+A^*}(v) - (|N_D^-(v) \setminus C|, |N_D^+(v) \setminus C|) \in \tau_C(v),$$

by definition of τ_C . Hence, also I' is a yes-instance.

Concerning the size of D' , observe that C contains at most $2s$ unsatisfied vertices and at most α satisfied vertices for each of the $(\Delta^*)^2$ possible types. Therefore,

$$|C| \leq 2s + (\Delta^* + 1)^2 \cdot \alpha \leq 2s + (\Delta^*)^2 \cdot 2s(\Delta_D + 1).$$

Since $\Delta_D \leq \Delta^*$, we obtain a problem kernel with $O(s(\Delta^*)^3)$ vertices. The overall running time is in $O(m + |\tau| + r^2)$. \square \square

The goal now is to use our framework (Theorem 11) to transfer the polynomial-size problem kernel with respect to (s, Δ^*) to a polynomial-size problem kernel with respect to Δ^* alone. To this end, we show that the corresponding number problem #DDCONC (which is the special case of #DDCONSEQC without the sequence property II) is polynomial-time solvable.

NUMBERS ONLY DIGRAPH DEGREE CONSTRAINT COMPLETION (#DDCONC)

Input: A sequence $\sigma = (c_1, d_1), \dots, (c_n, d_n)$ of n nonnegative integer tuples, a positive integer s , and a “tuple list function” $\tau: \{1, \dots, n\} \rightarrow 2^{\{0, \dots, r\}^2}$.

Question: Is there a sequence $\sigma' = (c'_1, d'_1), \dots, (c'_n, d'_n)$ such that $\sum_{i=1}^n c'_i - c_i = \sum_{i=1}^n d'_i - d_i = s$, and $c_i \leq c'_i$, $d_i \leq d'_i$, and $(c'_i, d'_i) \in \tau(i)$ for all $1 \leq i \leq n$?

#DDCONC can be solved in pseudo-polynomial time by a dynamic programming algorithm. Note that pseudo-polynomial time is sufficient for our purposes since all occurring

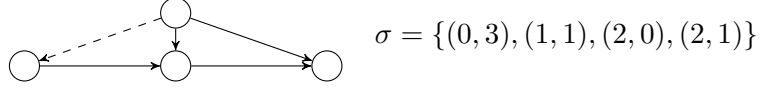


Figure 3: Example instance of DDSEQC. Inserting the dashed arc in the input digraph (solid arcs) with degree sequence $\{(0, 1), (0, 2), (2, 0), (2, 1)\}$ yields a digraph with the given target sequence σ .

numbers will be bounded by $O(n^2)$ when creating the $\#DDCONC$ instance from the given $DDCONC$ instance. (In fact, we conjecture that $\#DDCONC$ is weakly NP-hard and a reduction from PARTITION should be possible as in the case for $\#DDA$ in Section 4.3, Theorem 23.)

Lemma 14. $\#DDCONC$ is solvable in $O(n(sr)^2)$ time.

Proof. Let $I := ((c_1, d_1), \dots, (c_n, d_n), s, \tau)$ be an instance of $\#DDCONC$. We solve I using a modified version of the dynamic programming algorithm for NCE due to Froese et al. [14, Lemma 2]. To this end, we define the Boolean table $M[i, j, l]$ for $i \in \{1 \dots, n\}$, $j, l \in \{0, \dots, s\}$, where $M[i, j, l] = \text{true}$ if and only if there exist tuples $(c'_1, d'_1), \dots, (c'_i, d'_i)$ with $c'_p \geq c_p$, $d'_p \geq d_p$ and $(c'_p, d'_p) \in \tau(p)$ for all $p \in \{1, \dots, i\}$ such that $\sum_{p=1}^i (c'_p - c_p) = j$ and $\sum_{p=1}^i (d'_p - d_p) = l$. Thus, I is a yes-instance if $M[n, s, s] = \text{true}$. We compute M based on the recurrence where we essentially consider all possibilities to fix the i -th tuple and recurse:

$$M[i, j, l] = \text{true} \iff \exists (c'_i, d'_i) \in \tau(i) : (c'_i \geq c_i) \wedge (d'_i \geq d_i) \wedge M[i-1, j - (c'_i - c_i), l - (d'_i - d_i)],$$

where we set

$$M[1, j, l] := \begin{cases} \text{true}, & \text{if } (c_1 + j, d_1 + l) \in \tau(1), \\ \text{false}, & \text{otherwise.} \end{cases}$$

The size of M is in $O(ns^2)$. A single entry can be computed in $O(r^2)$ time. \square \square

Combining Theorem 13 and Lemma 14 yields the following corollary of Theorem 11.

Corollary 15. $DDCONC$ admits a problem kernel containing $O((\Delta^*)^5) \subseteq O(r^5)$ vertices. It is computable in $O(m + ns^3r^2)$ time.

4.2 Digraph Degree Sequence Completion

In this section, we investigate the NP-hard special case of $DDCONSEQC^3$ where τ does not restrict the allowed degree of any vertex and Π is fulfilled by exactly one specific degree sequence σ (see Figure 3 for an example). The undirected problem variant is studied by Golovach and Mertzios [19].

DIGRAPH DEGREE SEQUENCE COMPLETION (DDSEQC)

Input: A digraph $D = (V, A)$, a digraph degree sequence σ containing $|V|$ integer tuples.

Question: Is it possible to obtain a digraph D' by inserting arcs in D such that $\sigma(D') = \sigma$?

³Although not stated explicitly, the NP-hardness follows from the proof of Theorem 3.2 of the Bachelor thesis of Millani [33] (online available) as the construction therein allows for only one feasible target degree sequence.

For DDSEQC, the parameter Δ^* is by definition equal to Δ_σ . Moreover, note that the number s of arcs to insert (if possible) is determined by the target sequence σ by $s := \sum_{(c,d) \in \sigma} c - \sum_{v \in V(D)} \deg_D^-(v)$. We henceforth assume that

$$s = \sum_{(c,d) \in \sigma} c - \sum_{v \in V(D)} \deg_D^-(v) = \sum_{(c,d) \in \sigma} d - \sum_{v \in V(D)} \deg_D^+(v) \geq 0$$

holds since otherwise we have a trivial no-instance.

Since deciding Π (that is, deciding whether $\sigma(D') = \sigma$) can be done in polynomial time, we immediately obtain fixed-parameter tractability of DDSEQC with respect to (s, Δ_D) due to Theorem 6. We further strengthen this result by developing a polynomial-size problem kernel for DDSEQC with respect to (s, Δ_σ) . The kernelization is inspired by the $O(s\Delta_\sigma^2)$ -vertex problem kernel for the undirected problem by Golovach and Mertzios [19]. The main idea is to only keep the vertices of a $2s(\Delta_D + 1)$ -block set (see Definition 1) together with some additional “dummy” vertices and to adjust the digraph degree sequence σ properly.

Theorem 16. *DDSEQC admits a problem kernel containing $O(s\Delta_\sigma^3)$ vertices computable in $O(n + m + \Delta_\sigma^2)$ time.*

Proof. Let (D, σ) be a DDSEQC instance. Clearly, since we are only allowed to insert arcs in the digraph D , we can never decrease the in- or outdegree of any vertex. Hence, if $\Delta_D^- > \Delta_\sigma^-$ or $\Delta_D^+ > \Delta_\sigma^+$, then we return a trivial no-instance. Otherwise, we know that $\Delta_D \leq \Delta_\sigma$. Moreover, since inserting one arc can change the degrees of at most two vertices, it also holds $\lambda_D(\deg_D(v)) \leq \lambda_\sigma(\deg_D(v)) + 2s$ for each $v \in V(D)$.

We now compute a $2s(\Delta_D + 1)$ -block set C (see Definition 1) in $O(n + m + \Delta_D^2)$ time (Lemma 2) and return the instance (D', σ') which is defined as follows. The digraph D' is constructed from D by the following steps:

- Delete all vertices of $V(D) \setminus C$.
- Add $h := \Delta_\sigma + 2$ new vertices $W := \{w_1, \dots, w_h\}$ and insert all arcs W^2 .
- For each $v \in C$ such that the number $r_v^- := |N_D^-(v) \setminus C|$ of inneighbors in $V(D) \setminus C$ is at least one, insert the arcs $\{(w_i, v) \mid 1 \leq i \leq r_v^-\}$.
- For each $v \in C$ such that the number $r_v^+ := |N_D^+(v) \setminus C|$ of outneighbors in $V(D) \setminus C$ is at least one, insert the arcs $\{(v, w_i) \mid 1 \leq i \leq r_v^+\}$.

The digraph D' can be constructed in $O(n + m + \Delta_\sigma^2)$ time. Observe that $\deg_{D'}^-(w_i) \geq \Delta_\sigma + 1$ and $\deg_{D'}^+(w_i) \geq \Delta_\sigma + 1$ holds for all $i \in \{1, \dots, h\}$, and that $\deg_{D'}(v) = \deg_D(v) \leq \Delta_\sigma$ holds for all $v \in C$. The number of vertices in D' equals $|C| + h$. Note that C contains at most $2s(\Delta_D + 1)$ vertices of each of the $(\Delta_D + 1)^2$ possible vertex degrees in D . Thus, D' contains $O(s\Delta_\sigma^3)$ vertices.

The digraph degree sequence σ' is constructed from σ as follows:

- For each vertex $v \in V(D) \setminus C$ that was removed from D , remove a copy of the tuple $\deg_D(v)$ from σ .
- For each $i \in \{1, \dots, h\}$, add the tuple $\deg_{D'}(w_i)$.

Note that this construction is well-defined, that is, we can always apply the first step and remove a copy of $\deg_D(v)$ from σ since we remove at most

$$\lambda_D(\deg_D(v) - 2s(\Delta_D + 1)) < \lambda_D(\deg_D(v)) - 2s \leq \lambda_\sigma(\deg_D(v))$$

copies. The construction of σ' can be done in $O(n)$ time. Hence, the overall running time of computing the problem kernel is in $O(n + m + \Delta_\sigma^2)$.

It remains to show that (D', σ') is a yes-instance if and only if (D, σ) is a yes-instance. Assume first that (D, σ) is a yes-instance. We know from Lemma 5 that there exists a solution $A^* \subseteq C^2$ with $\sigma(D + A^*) = \sigma$. Using

$$\begin{aligned} \forall v \in V(D) \setminus C : \deg_{D+A^*}(v) &= \deg_D(v), \\ \forall v \in C : \deg_{D'+A^*}(v) &= \deg_{D+A^*}(v), \text{ and} \\ \forall w_i \in W : \deg_{D'+A^*}(w_i) &= \deg_{D'}(w_i), \end{aligned}$$

it is then easy to verify that $\sigma(D' + A^*) = \sigma'$, and thus, (D', σ', s) is a yes-instance.

Conversely, let $A' \subseteq V(D')^2$ be a solution for (D', σ') with $\sigma(D' + A') = \sigma'$. We claim that $A' \subseteq C^2$, that is, A' does not contain an arc incident to a vertex in W . To see this, recall that by construction

$$\begin{aligned} \deg_{D'}^-(w_1) = \Delta_{\sigma'}^- &\geq \dots \geq \deg_{D'}^-(w_h) \geq \Delta_\sigma + 1 > \deg_{D'}^-(v) \text{ and} \\ \deg_{D'}^+(w_1) = \Delta_{\sigma'}^+ &\geq \dots \geq \deg_{D'}^+(w_h) \geq \Delta_\sigma + 1 > \deg_{D'}^+(v) \end{aligned}$$

hold for all $v \in C$. That is, $\deg_{D'}(w_1) = (\Delta_{\sigma'}^-, \Delta_{\sigma'}^+)$, and thus a solution must not insert arcs incident to w_1 . It follows that $\deg_{D'+A'}(w_1) = \deg_{D'}(w_1)$. This recursively also holds for w_2, \dots, w_h and thus, we have $\deg_{D'+A'}(w_i) = \deg_{D'}(w_i)$ for all $w_i \in W$. Hence, A' does not contain any arcs incident to vertices in W , that is, $A' \subseteq C^2$. Thus, we can derive

$$\begin{aligned} \forall v \in C : \deg_{D'+A'}(v) &= \deg_{D+A'}(v), \text{ and} \\ \forall v \in V(D) \setminus C : \deg_{D+A'}(v) &= \deg_D(v). \end{aligned}$$

It is now straightforward to check that $\sigma(D + A') = \sigma$. □ □

To apply our framework and derive a polynomial-size problem kernel with respect to Δ_σ for DDSEQC, we define a corresponding number problem and show its polynomial-time solvability. The number problem #DDSEQC is the special case of #DDCONSEQC asking for the specific target sequence σ .

NUMBERS ONLY DIGRAPH DEGREE SEQUENCE COMPLETION (#DDSEQC)

Input: Two multisets $\sigma = \{(c_1, d_1), \dots, (c_n, d_n)\}$ and $\phi = \{(c'_1, d'_1), \dots, (c'_n, d'_n)\}$ containing n nonnegative integer tuples.

Question: Is there a bijection $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that $c_i \leq c'_{\pi(i)}$, and $d_i \leq d'_{\pi(i)}$, for all $1 \leq i \leq n$?

#DDSEQC can be solved in polynomial time by finding perfect matchings in an auxiliary graph.

Lemma 17. #DDSEQC is solvable in $O(n^{2.5})$ time.

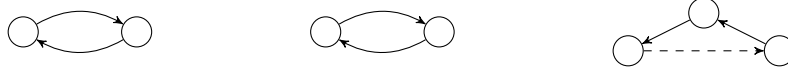


Figure 4: Example instance of DDA. The input digraph with three components (solid arcs) is 1-anonymous since there is only one vertex with degree $(0, 1)$. By inserting the dashed arc, the digraph becomes 7-anonymous since all vertices have degree $(1, 1)$.

Proof. We show how to solve the problem by computing a perfect matching in a bipartite graph. Let $(\{(c_1, d_1), \dots, (c_n, d_n)\}, \{(c'_1, d'_1), \dots, (c'_n, d'_n)\})$ be a $\#DDSEQC$ instance. We construct an undirected bipartite graph $G := (V \cup W, E)$. For each $i \in \{1, \dots, n\}$, there is a vertex $v_i \in V$ corresponding to the tuple (c_i, d_i) , and a vertex $w_i \in W$ corresponding to (c'_i, d'_i) . For each $i, j \in \{1, \dots, n\}$, $i \neq j$, the edge $\{v_i, w_j\}$ is in E if and only if $c'_j \geq c_i$ and $d'_j \geq d_i$ hold. The graph G can be computed in $O(n^2)$ time. Note that a perfect matching in G defines a bijection that satisfies the condition in the problem definition. Hence, we can solve a $\#DDSEQC$ instance by computing a perfect matching in a bipartite graph, which can be done in $O(|E|\sqrt{|V \cup W|}) = O(n^{2.5})$ time [25]. \square \square

Combining Theorem 16 and Lemma 17 yields the following corollary of Theorem 11.

Corollary 18. *DDSEQC admits a problem kernel containing $O(\Delta_\sigma^5)$ vertices. It is computable in $O(sn^{2.5})$ -time.*

4.3 Degree Anonymity

We extend the definition of DEGREE ANONYMITY in undirected graphs due to Liu and Terzi [31] to digraphs and obtain the following NP-hard problem [33] (Figure 4 presents an example):

DIGRAPH DEGREE ANONYMITY (DDA)

Input: A digraph $D = (V, A)$ and two positive integers k and s .

Question: Is it possible to obtain a digraph D' by inserting at most s arcs in D such that D' is k -anonymous, that is, for every vertex $v \in V$ there are at least $k - 1$ other vertices in D' with degree $\deg_{D'}(v)$?

The (parameterized) complexity as well as the (in-)approximability of the undirected version called DEGREE ANONYMITY are well-studied [4, 7, 24]. There also exist many heuristic approaches to solve the undirected version [5, 23]. Notably, our generic approach shown in Section 3.2 originates from a heuristic of Liu and Terzi [31] for DEGREE ANONYMITY. Later, Hartung et al. [24] used this heuristic to prove that “large” solutions of DEGREE ANONYMITY can be found in polynomial time and Froese et al. [14] extended this approach to a more general class of problems. The property Π (that is, k -anonymity) can clearly be checked for a given input digraph degree sequence in polynomial time. Hence, Theorem 6 yields fixed-parameter tractability of DDA with respect to (s, Δ_D) . Again, we develop a polynomial-size problem kernel with respect to (s, Δ_D) . Somewhat surprisingly, we cannot transfer this problem kernel to a problem kernel with respect to Δ^* since we are not able to solve the corresponding number problem in polynomial time. In fact, we will show that it is at least weakly NP-hard.

To start with, we give a problem kernel based on Lemma 5 in a similar fashion as in the proof of Theorem 16. More precisely, by Lemma 5 we know that we only need to keep a $2s(\Delta_D + 1)$ -block set C , that is, $2s(\Delta_D + 1)$ arbitrary vertices of each block. Note that

deleting all vertices that are not in C changes the degrees of the vertices in C . We repair this in a similar way as in the problem kernel stated in Theorem 16: After deleting the vertices that are not in C , we add vertices adjacent to the vertices in C in such a way that the vertices in C keep their original degrees. Denoting the set of newly added vertices by P , we also need to “separate” the vertices in P from the vertices in C so that their degrees do not interfere in the target degree sequence. We do this, similarly as in the proof of Theorem 16, by increasing the degrees of all vertices in P to at least $\Delta_D + s + 1$. Furthermore, we need to ensure that a solution in the new instance does not insert arcs between vertices in C and vertices in P since we cannot map such solutions back to solutions for the original instance. Solving this issue, however, is not as simple as for DDSEQC and requires some adjustment of the actual number of vertices we keep. As a result, we will prove that if there is a solution inserting arcs between C and P , then there is also a solution not inserting such arcs (Lemma 19).

Another adjustment concerns the anonymity level k : If k is large, then we need to shrink it since otherwise we would always create no-instances. The general idea is to keep the “distance to size k ”, meaning that if in the original instance some block contains $k + x$ vertices for some $x \in \{-2s, \dots, 2s\}$, then in the new instance this block should contain $k' + x$ vertices where k' is the new anonymity level. The reason for the specific range of values for x between $-2s$ and $2s$ is that if some block has size larger than $k + 2s$ for example, then, after inserting s arcs, this block will still be of size larger than k . Similarly, if a block contains less than $k - 2s$ vertices, then after inserting s arcs it will contain less than k vertices and it will violate the k -anonymity constraint unless it is empty. Hence, the interesting cases for x are between $-2s$ and $2s$. In order to ensure that there is a solution not inserting arcs between C and P , we need to increase this range from $-2s$ to $4s$, see the proof of Lemma 19 for further details.

In the following, we describe the details of our kernelization algorithm, see Algorithm 1 for the pseudocode. Observe that our general approach is a non-obvious adaption of the polynomial-size problem kernel for the undirected DEGREE ANONYMITY problem by Hartung et al. [24].

Lemma 19. *Let (D, k, s) be an instance of DDA and let (D', k', s) be the instance computed by Algorithm 1, where $P := P_{\text{in}} \cup P_{\text{out}}$ is the set of newly added vertices. If there is a solution $S \subseteq V(D')^2$ with $|S| \leq s$, then there is also a solution $S' \subseteq V(D')^2$ with $|S'| \leq |S|$ such that $V(S') \cap P = \emptyset$.*

Proof. Let $S \subseteq V(D')^2$ be a solution for (D', k', s) such that $V(S') \cap P \neq \emptyset$. We construct a new solution $S' \subseteq V(D')^2$ such that $|S'| \leq |S|$ and $V(S') \cap P = \emptyset$. The idea is to replace the endpoints of arcs that are in P by new endpoints from one “large” block (of size at least $\beta + 2s$) in C . To this end, observe that if $V(D) \leq (\Delta_D + 1)^2(\beta + 2s)$, then Algorithm 1 returns the original instance (see Algorithm 1) and we are done. Hence, there is at least one block $B_{D'}(t)$ for some $t \in \sigma(D')$ of size at least $\beta + 2s$ since there are at most $(\Delta_D + 1)^2$ blocks. We will use vertices in $B_{D'}(t)$ as a replacement for the vertices in P within the arcs of S .

We now construct S' . To this end, initialize $S' := S \cap C^2$ and insert further arcs in the following way. First, consider those arcs in S that have exactly one endpoint in P . For each arc $(u, v) \in S$ with $u \in C$ and $v \in P$, insert the arc (u, w) in S' where $w \in B_{D'}(t)$ such that w is not incident to any arc in S' and not an outneighbor of u . Since $|B_{D'}(t)| \geq \beta + 2s = (\Delta_D + 3)2s$ and $|S'| \leq s$, it follows that $B_{D'}(t)$ contains such a vertex w . Similarly, for each arc $(v, u) \in S$ with $u \in C$ and $v \in P$, insert the arc (w, u) in S' where $w \in B_{D'}(t)$ is a vertex not incident

Algorithm 1: The pseudocode of the algorithm computing a polynomial-size kernel with respect to (s, Δ_D) for DDA.

Input: A digraph $D = (V, A)$ and integers $k, s \in \mathbb{N}$.

Output: A digraph D' and integers $k', s \in \mathbb{N}$.

```

1 if  $|V| \leq (\Delta_D + 1)^2(\beta + 2s)$  then                                //  $\beta$  is defined as  $\beta := (\Delta_D + 2)2s$ 
2   return  $(D, k, s)$ 
3  $k' \leftarrow \min\{k, \beta\}$ 
4  $C \leftarrow \emptyset$ 
5 foreach distinct tuple  $t$  occurring in  $\sigma(D)$  do
6   if  $2s < |B_D(t)| < k - 2s$  then                                // insufficient budget for  $B_D(t)$ 
7     return trivial no-instance
8   if  $k \leq \beta$  then                                                // determine number of retained vertices
9      $x \leftarrow \min\{|B_D(t)|, \beta + 2s\}$                             // keep at most  $\beta + 2s$  vertices
10  else if  $|B_D(t)| \leq 2s$  then                                    // “small” block
11     $x \leftarrow |B_D(t)|$                                             // keep all vertices (“distance to size zero”)
12  else                                                            // “large” block and  $k' = \beta$ 
13     $x \leftarrow k' + \min\{2s, (|B_D(t)| - k)\}$                     // keep “distance to size  $k$ ”
14  add  $x$  arbitrary vertices from  $B_D(t)$  to  $C$ 
15  $D' \leftarrow D[C]$ 
16 foreach  $v \in C$  do                                              // insert new vertices to preserve degrees of vertices in  $C$ 
17   add  $\deg_D^+(v) - \deg_{D'}^+(v)$  many vertices with an incoming arc from  $v$  to  $D'$ 
18   add  $\deg_D^-(v) - \deg_{D'}^-(v)$  many vertices with an outgoing arc to  $v$  to  $D'$ 
19 let  $P_{\text{in}}$  be the set of vertices added in Algorithm 1              //  $\forall v \in P_{\text{in}}: \deg_{D'}(v) = (1, 0)$ 
20 let  $P_{\text{out}}$  be the set of vertices added in Algorithm 1            //  $\forall v \in P_{\text{out}}: \deg_{D'}(v) = (0, 1)$ 
21 while  $\min\{|P_{\text{in}}|, |P_{\text{out}}|\} < \max\{\Delta_D + s + 1, k'\}$  do
22   add a new vertex  $v$  to  $P_{\text{in}}$ 
23   add a new vertex  $u$  to  $P_{\text{out}}$ 
24   insert the arc  $(u, v)$  in  $D'$ 
25 insert all arcs  $P_{\text{in}}^2$  and  $P_{\text{out}}^2$  in  $D'$                         // ensure high degree difference from vertices in  $C$ 
26 insert all arcs from  $P_{\text{in}} \times P_{\text{out}}$  in  $D'$                         // separate  $P_{\text{in}}$  from  $P_{\text{out}}$ 
27 return  $(D', k', s)$ 

```

to any arc in S' and not an inneighbor of u . Again, due to the size of $B_{D'}(t)$, such a vertex exists.

Second, consider those arcs in S having both endpoints in P . For each arc $(u, v) \in S$ with $u, v \in P$, insert the arc (u', v') in S' where $u', v' \in B_{D'}(t)$ such that neither u' nor v' is incident to any arc in S' and $(u', v') \notin A(D')$. Since $|B_{D'}(t)| \geq \beta + 2s = (\Delta_D + 3)2s$ and $|S'| \leq s$, it follows that these vertices u' and v' exist. Observe that after all these modifications, there are still at least β vertices left in $B_{D'}(t)$.

Clearly, we have $|S'| \leq |S|$. It remains to prove that $D' + S'$ is k' -anonymous. To this end, observe that since the outdegree of each vertex in P_{in} is at least $|P_{\text{out}}| - 1 \geq \Delta_D + s + 1$ (see Algorithm 1) larger than the outdegree of any vertex in P_{out} , it follows that the vertices in P which are incident to an arc in S end up in blocks of $D' + S$ that are empty in D' . Thus, at least k' vertices in P are the head of an arc in S and at least k' vertices in P are the tail of an arc in S . Hence, we used at least k' vertices from $B_{D'}(t)$ as a replacement in S' and thus the blocks $B_{D'+S'}(t + (1, 0))$ and $B_{D'+S'}(t + (0, 1))$ contain at least k' vertices. Furthermore,

all other vertices in C have the same degree in $D' + S$ and in $D' + S'$ and the vertices in P are not incident to any arc in S' . Since S was a solution, it follows that also $D' + S'$ is k' -anonymous. \square

We remark that parts of the proof of Lemma 19 are an adaption of the proof of the corresponding lemma in the undirected case [24, Lemma 6].

Theorem 20. *DDA admits a problem kernel containing $O(\Delta_D^5 s)$ vertices. It is computable in $O(\Delta_D^{10} s^2 + \Delta_D^3 s n)$ time.*

Proof. We use Algorithm 1 to compute the problem kernel. The correctness of the kernelization follows from the following two lemmas. Their proofs are, however, adaptations of the corresponding undirected counterparts [24, Lemmas 7 and 8]

Let $D = (V, A)$ be a digraph and $k \in \mathbb{N}$. An arc set $S \subseteq V^2$ is called k -insertion set for D , if $D + S$ is k -anonymous.

Lemma 21. *If the instance (D', k', s) constructed by Algorithm 1 is a yes-instance, then (D, k, s) is a yes-instance.*

of Lemma 21. First, observe that if $k \leq \beta$, then $k' = k$ and each k -insertion set for D' is a k -insertion set for D as all blocks with less than $\beta + 2s$ vertices remain unchanged. Hence, it remains to consider the case that $k > \beta$ and thus $k' = \beta$.

Let S' be an arc set such that $|S'| \leq s$ and $D' + S'$ is k' -anonymous. By Lemma 19, we can assume that each arc in S' has both endpoints in C . We show that $D + S'$ is k -anonymous, that is, for each block $B_{D+S'}(t)$ we have $|B_{D+S'}(t)| \geq k$ or $|B_{D+S'}(t)| = 0$. To this end, we distinguish two cases on whether the corresponding block in $D' + S'$ is empty or contains at least k' vertices.

First, consider the case $|B_{D'+S'}(t)| = 0$. Since it holds that $|S'| \leq s$, it follows that $|B_{D'}(t)| \leq 2s$. By Algorithms 1 and 1, it follows that D and D' contain the same vertices of degree t , that is, $B_{D'}(t) = B_D(t)$. Hence, we have $|B_{D+S'}(t)| = 0$.

Second, consider the case $|B_{D'+S'}(t)| \geq k$. If $|B_D(t)| \geq k + 2s$, then it clearly holds that $|B_{D+S'}(t)| \geq k$ and we are done. Otherwise, by Algorithm 1, we have $|B_D(t)| - k = |B_{D'}(t)| - k'$. Since S' only contains arcs with both endpoints in C , it follows that by inserting S , the same vertices will be added and removed from $B_D(t)$ and $B_{D'}(t)$, that is, $|B_{D+S'}(t)| - k = |B_{D'+S'}(t)| - k'$. Since $|B_{D'+S'}(t)| \geq k'$ it follows that $|B_{D+S'}(t)| \geq k$. Thus, $D + S'$ is k -anonymous and (D, k, s) is a yes-instance. (Proof of Lemma 21) \square

Lemma 22. *If (D, k, s) is a yes-instance, then the instance (D', k', s) constructed by Algorithm 1 is a yes-instance.*

Proof. Observe that in the instance (D', k', s) constructed by Algorithm 1, for each degree $t \in \mathbb{N}^2$, we have that either $B_D(t) = B_{D'}(t)$ (in case that $B_D(t)$ contains few vertices, see Algorithms 1 and 1) or $B_{D'}(t) \subseteq B_D(t)$ contains at least $\beta - 2s$ vertices ($|B_{D'}(t)| \geq \beta - 2s = (\Delta_D + 1)2s$, see Algorithms 1 to 1). Thus, D' contains a $(\Delta_D + 1)2s$ -block set. Since (D, k, s) is a yes-instance, it follows from Lemma 5 that there is a k -insertion set S of size at most s for D such that $S \subseteq C^2$.

We next show that $D' + S$ is k' -anonymous, and hence, (D', k', s) is a yes-instance. First, consider the case that $k \leq \beta$ and thus $k' = k$. Observe that every block $B_{D'}(t)$ containing at least $k + 2s$ vertices contains at least $k = k'$ vertices in $D' + S$. For every block $B_{D'}(t)$

containing less than $k + 2s$ vertices it holds that $B_{D'}(t) = B_D(t)$ (see Algorithm 1). Thus, $|B_{D+S}(t)| = |B_{D'+S}(t)|$ and therefore $B_{D'+S}(t)$ fulfills the k -anonymity requirement.

Second, consider the case that $k > \beta$ and thus $k' = \beta$. Let $B_{D'}(t)$ be some block of D' . We show that $|B_{D'+S}(t)| = 0$ or $|B_{D'+S}(t)| \geq k'$. If $|B_{D'}(t)| \leq 2s$, then $B_{D'}(t) = B_D(t)$ (see Algorithm 1). Hence, $B_{D'+S}(t) = B_{D+S}(t) = 0$ since $k > \beta > 4s$. If $|B_{D'}(t)| > 2s$, then $|B_D(t)| > k - 2s$ (see Algorithm 1) and thus $|B_{D'}(t)| = \beta + \min\{2s, (|B_D(t)| - k)\}$ (see Algorithm 1). Observe that $|B_{D'+S}(t)| - |B_{D'}(t)| = |B_{D+S}(t)| - |B_D(t)|$, and thus,

$$|B_{D'+S}(t)| = (|B_{D+S}(t)| - |B_D(t)|) + |B_{D'}(t)|. \quad (2)$$

Since $|S| \leq s$, we have $|B_{D+S}(t)| - |B_D(t)| \geq -2s$. We now distinguish the two cases $|B_D(t)| - k \geq 2s$ and $|B_D(t)| - k < 2s$. In the first case, it follows that $|B_{D'}(t)| = \beta + 2s$ and from eq. (2) it follows

$$|B_{D'+S}(t)| \geq -2s + \beta + 2s = \beta = k'.$$

In the second case, it follows that $|B_{D'}(t)| = \beta + |B_D(t)| - k$ (see Algorithm 1). Observe that $|B_{D+S}(t)| \geq k$ since $|B_D(t)| > k - 2s$. From eq. (2) we conclude that

$$|B_{D'+S}(t)| \geq k - |B_D(t)| + \beta + |B_D(t)| - k = \beta = k'.$$

(Proof of Lemma 22) \square

The size of the kernel can be seen as follows: For each of the at most $(\Delta_D + 1)^2$ different blocks in the input graph D , the algorithm keeps at most $\beta + 2s = (\Delta_D + 3)2s$ vertices in the set C (see Algorithms 1 to 1). Thus, $|C| \in O(\Delta_D^3 s)$. The number of newly added vertices in Algorithms 1 to 1 is at most $\max\{\Delta_D^2 \cdot |C|, k', \Delta_D + s + 1\}$. Hence, $|P| \in O(\Delta_D^5 s)$ and thus the instance produced by Algorithm 1 contains at most $O(\Delta_D^5 s)$ vertices.

The running time can be seen as follows: Using bucket sort, one can lexicographically sort the n vertices by degree in $O(n)$ time. Furthermore, in the same time one can create $(\Delta_D + 1)^2$ lists—each list containing the vertices of some degree $t \in \mathbb{N}^2$. Then, the selection of the $O(\Delta_D^3 s)$ vertices of C can be done in $O(\Delta_D^3 sn)$ time. Clearly, inserting the vertices in P can be done in $O(\Delta_D^5 s)$ time. Finally, inserting the arcs between the vertices in P (Algorithms 1 and 1) takes $O(\Delta_D^{10} s^2)$ time. \square

In contrast to both number problems in Sections 4.1 and 4.2, we were unable to find a polynomial-time algorithm for the number problem for DDA, which is the special case of #DDCONSEQC asking for a k -anonymous target sequence.

NUMBERS ONLY DIGRAPH DEGREE ANONYMITY (#DDA)

Input: A sequence $\sigma = (c_1, d_1), \dots, (c_n, d_n)$ of n nonnegative integer tuples, two positive integers s and k .

Question: Is there a sequence $\sigma' = (c'_1, d'_1), \dots, (c'_n, d'_n)$ such that

- (i) $\sum_{i=1}^n c'_i - c_i = \sum_{i=1}^n d'_i - d_i = s$,
- (ii) $c_i \leq c'_i$, and $d_i \leq d'_i$ for all $1 \leq i \leq n$, and
- (iii) each tuple in σ' appears at least k times?

We can show that #DDA is weakly NP-hard by a polynomial-time many-one reduction from PARTITION.

PARTITION

Input: A multiset $A = \{a_1, \dots, a_n\}$ of positive integers that sum up to $2B$.

Question: Is there a subset $A' \subseteq A$ whose elements sum up to B ?

Theorem 23. *#DDA is (weakly) NP-hard even if $k = 2$.*

Proof. Given a multiset $A = \{a_1, \dots, a_n\}$, observe that we can assume without loss of generality that each integer in A is smaller than B (otherwise we could solve the instance in polynomial time).

We create the following #DDA-instance with $s := B$, $k := 2$, and the sequence σ containing the following tuples. For each $a_i \in A$ create five tuples: one tuple x_i of type $(2B(i+1) - a_i, 0)$, one block X_i that contains two tuples of type $(2B(i+1), 0)$, and one block X'_i that contains two tuples of type $(2B(i+1) - a_i, a_i)$. This completes the construction.

We show that there is a subset $A' \subset A$ whose elements sum up to exactly B if and only if there is a sequence $\sigma' = (c'_1, d'_1), \dots, (c'_n, d'_n)$ that fulfills Conditions (i)–(iii) of #DDA.

First, assume that there is some $A' \subset A$ and $\sum_{a \in A'} a = B$. Then, we obtain the desired sequence σ' by first copying σ and changing x_i as follows: For each $a_i \in A'$ change the tuple x_i from type $(2B(i+1) - a_i, 0)$ to type $(2B(i+1), 0)$ and for each $a_i \notin A'$ change the tuple x_i from type $(2B(i+1) - a_i, 0)$ to type $(2B(i+1) - a_i, a_i)$. It is not hard to verify that this σ' is indeed a solution: For Condition (i), observe that $\sum_{i=1}^n (c'_i - c_i) = s = B = \sum_{i=1}^n (d'_i - d_i)$ since the elements in A' as well as the elements in $A \setminus A'$ sum up to B . Condition (ii) is clearly ensured by construction of σ' . For Condition (iii), note that in sequence σ' block X_i contains either $k = 2$ tuples (if $a_i \notin A'$) or $k + 1$ tuples (if $a_i \in A'$) and, analogously, note that block X'_i contains either $k = 2$ tuples (if $a_i \in A'$) or $k + 1$ tuples (if $a_i \notin A'$); σ' contains no further tuples.

Second, assume that there is a sequence $\sigma' = (c'_1, d'_1), \dots, (c'_n, d'_n)$ that is a solution for our constructed #DDA instance. First note that σ' does not differ from σ “a lot” in the following sense. Since $s = B$ and $k = 2$, in sequence σ' the first component and the second component of all tuples can in total be increased by at most B , respectively. Next, observe that each tuple x_i must either be of type $(2B(i+1), 0)$ or of type $(2B(i+1) - a_i, a_i)$, since every other tuple is too far away (recall that $a < B$ for all $a \in A$). This means that each tuple x_i contributes with a_i to the total sum over the differences in either the first component ($\sum_{i=1}^n (c'_i - c_i)$), or the second component ($\sum_{i=1}^n (d'_i - d_i)$). Since $\sum_{a \in A} a = 2B$, it follows that the tuples x_i require at least a budget of B in either the first or the second component. Let $A' := \{a_i \mid x_i \text{ is of type } (2B(i+1), 0) \text{ in } \sigma'\}$. We show that $\sum_{a \in A'} a = B$. Assume towards a contradiction that $\sum_{a \in A'} a \neq B$. Since $\sum_{i=1}^n (c'_i - c_i) = \sum_{a \in A'} a$ and $\sum_{i=1}^n (d'_i - d_i) = \sum_{a \notin A'} a$, either $\sum_{i=1}^n (c'_i - c_i)$ or $\sum_{i=1}^n (d'_i - d_i)$ would be greater than B —a contradiction to our budget. \square \square

Note that the hardness from Theorem 23 does not translate to instances of #DDA originating from digraph degree sequences because in such instances all numbers in the input sequence σ and also in the output sequence σ' are bounded by $n - 1$ where n is the number of tuples in σ . Since there are pseudo-polynomial-time algorithms for PARTITION, Theorem 23 leaves open whether #DDA is strongly NP-hard or can be solved in polynomial time for instances originating from digraphs.

To again apply our framework (Theorem 10), we show that #DDA is at least fixed-parameter tractable with respect to the largest possible integer ξ in the output sequence. To

this end, we develop an integer linear program that contains at most $O(\xi^4)$ integer variables and apply the a famous result due to Lenstra [29].

Theorem 24. *#DDA is fixed-parameter tractable with respect to the largest possible integer ξ in the output sequence.*

Proof. Let (σ, s, k) be an instance of #DDA. The key idea is that knowing how many tuples of type t in σ are transformed into a tuples of type t' in σ' for each pair $\{t, t'\}$ of tuples is sufficient to describe a solution of our #DDA instance. To this end, observe that there are at most $(\xi + 1)^2$ tuple blocks in σ and in σ' , respectively.

We describe an integer linear problem and create one variable $x_{t,t'}$ for each pair $t, t' \in \{0, \dots, \xi\}^2$ which denotes the number of tuples of type t in sequence σ that become tuples of type t' in sequence σ' . We further use the binary variables u_t for each $t \in \{0, \dots, \xi\}^2$ being 1 if and only if some tuple of type t is used in the solution, that is, there is at least one tuple of type t in σ' . We add a set of constraints ensuring that all tuples from σ appear in σ' :

$$\forall t \in \{0, \dots, \xi\}^2 : \sum_{t' \in \{0, \dots, \xi\}^2} x_{t,t'} = \lambda_\sigma(t).$$

Then, we ensure that (i) holds by:

$$\sum_{(t_1, t_2), (t'_1, t'_2) \in \{0, \dots, \xi\}^2} (t'_1 - t_1) \cdot x_{(t_1, t_2), (t'_1, t'_2)} = s$$

and by:

$$\sum_{(t_1, t_2), (t'_1, t'_2) \in \{0, \dots, \xi\}^2} (t'_2 - t_2) \cdot x_{(t_1, t_2), (t'_1, t'_2)} = s.$$

We ensure that (ii) holds by:

$$\forall (t_1, t_2), (t'_1, t'_2) \in \{0, \dots, \xi\}^2 \text{ with } t'_1 < t_1 \text{ or } t'_2 < t_2: x_{(t_1, t_2), (t'_1, t'_2)} = 0.$$

We ensure that (iii) holds by:

$$\forall t' \in \{0, \dots, \xi\}^2 : \sum_{t \in \{0, \dots, \xi\}^2} x_{t,t'} + k \cdot (1 - u_{t'}) \geq k.$$

Finally, we add the following constraint set to ensure consistency between the u_t and $x_{t,t'}$ variables:

$$\forall t' \in \{0, \dots, \xi\}^2 : \sum_{t \in \{0, \dots, \xi\}^2} x_{t,t'} \leq u_{t'} \cdot n.$$

Finally, fixed-parameter tractability follows by the famous result of Lenstra [29] (later improved by Frank and Tardos [13], Kannan [26]) that says that an ILP with ρ variables and ℓ input bits can be solved in $O(\rho^{2.5\rho+o(\rho)}\ell)$ time. \square \square

Combining Theorems 6, 10 and 24 yields fixed-parameter tractability for DDA with respect to Δ^* .

Corollary 25. *DIGRAPH DEGREE ANONYMITY is fixed-parameter tractable with respect to Δ^* .*

For undirected graphs, Hartung et al. [24] showed fixed-parameter tractability with respect to the maximum degree Δ_G of the input graph. This result was based on showing that $\Delta^* \in O(\Delta_G^2)$. For directed graphs, however, we can only show that $\Delta^* \leq 4k(\Delta_D + 2)^2$ implying fixed-parameter tractability with respect to (k, Δ_D) .

Lemma 26. *Let D be a digraph and let S be a minimum size arc set such that $D + S$ is k -anonymous. Then $\Delta_{D+S} \leq 4k(\Delta_D + 2)^2 + \Delta_D$.*

Proof. Let $D = (V, A)$ be a digraph with maximum degree Δ_D and let k be a positive integer. An arc set $S \subseteq V^2$ is called k -insertion set for D if $D + S$ is k -anonymous. Further, let $S \subseteq V^2$ be a minimum size k -insertion set. We will show that if $|V(S)| \geq 4k(\Delta_D + 2)^2$, then the maximum degree in $D + S$ is at most $\Delta_D + 2$, and if $|V(S)| < 4k(\Delta_D + 2)^2$, then the degree in $D + S$ is clearly at most $4k(\Delta_D + 2)^2 + \Delta_D$.

Now suppose that $|V(S)| \geq 4k(\Delta_D + 2)^2$ and assume towards a contradiction that $D + S$ has a maximum degree $\Delta_{D+S} > \Delta_D + 2$. We next construct a smaller k -insertion set S' in two steps. In the first step, we define for each vertex $v \in V$, a target degree $\tau(v) = (\tau^-(v), \tau^+(v))$ such that the following (and further conditions that are discussed later) holds:

- (a) $\deg_D^-(v) \leq \tau^-(v) \leq \Delta_D + 2$,
- (b) $\deg_D^+(v) \leq \tau^+(v) \leq \Delta_D + 2$, and
- (c) the multiset $\sigma(\tau) := \{\tau(v) \mid v \in V\}$ is k -anonymous, that is $\lambda_{\sigma(\tau)}(\tau(v)) \geq k$ for each $v \in V$.

As a second step, we use Lemma 7 to provide an arc set S' such that $\sigma(D + S') = \sigma(\tau)$. Since $\sigma(\tau)$ is k -anonymous, it follows that S' is a k -insertion set and we will show that $|S'| < |S|$.

We now give a detailed description of the two steps and start with defining the target degree function τ as follows

$$\tau(v) := (\min\{\Delta_D + 1, \deg_{D+S}^-(v)\}, \min\{\Delta_D + 1, \deg_{D+S}^+(v)\}). \quad (3)$$

Observe that τ satisfies the above three Conditions (a) to (c). Furthermore, we have $\sum_{v \in V} \deg_{D+S}^+(v) > \sum_{v \in V} \tau^+(v)$ since the maximum degree in $D + S$ is larger than $\Delta_D + 2$. If we can realize the target degrees τ with a k -insertion set S' , then it follows that $|S'| < |S|$.

To apply Lemma 7 with $\Delta_{D'} := \Delta_D + 2$, $x_i := \tau^+(v_i) - \deg_D^-(v_i)$ and $y_i := \tau^-(v_i) - \deg_D^+(v_i)$, we need to satisfy Conditions (I) to (IV) of Lemma 7. By assumption, $\Delta_{D'} = \Delta_D + 2 < \Delta_{D+S} \leq |V| - 1$ holds. Hence, Condition (I) is fulfilled. Moreover, $\tau^-(v) \leq \Delta_{D'}$ and $\tau^+(v) \leq \Delta_{D'}$ holds for all $v \in V$. Conditions (II) and (III) are thus also satisfied. However, we also need to ensure $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$ (Condition (IV)), that is, we need to ensure that τ changes the indegrees and outdegrees of the vertices in D by the same overall amount. This might not be true as we changed the indegrees and outdegrees independently. To overcome this problem, we subsequently adjust τ again.

Assume without loss of generality that compared to $D + S$ the target degree function τ reduced more indegrees than outdegrees, that is,

$$\sum_{v \in V} (\tau^+(v) - \deg_D^+(v)) > \sum_{v \in V} (\tau^-(v) - \deg_D^-(v)).$$

Denote by diff_τ the difference between the two sums, that is,

$$\begin{aligned}
\text{diff}_\tau &:= \sum_{v \in V} ((\tau^+(v) - \deg_D^+(v)) - (\tau^-(v) - \deg_D^-(v))) \\
&= \sum_{v \in V} (\tau^+(v) - \tau^-(v) + \deg_D^-(v) - \deg_D^+(v)) \\
&= \sum_{v \in V} (\tau^+(v) - \tau^-(v)) + \sum_{v \in V} (\deg_D^-(v) - \deg_D^+(v)) \\
&= \sum_{v \in V} (\tau^+(v) - \tau^-(v)).
\end{aligned}$$

Further, denote by $B_\tau(\tau(v))$ the *block of v in τ* , that is the set vertices having the same target degree as v . In the final adjustment of τ we need diff_τ to be at least k and at most $3k$. Hence, if $\text{diff}_\tau < k$, then we adjust τ as follows: Pick an arbitrary vertex v such that the outdegree of v in $D + S$ is larger than $\Delta_D + 1$. Observe that such a vertex must exist: We assumed to reduce the indegrees more than the outdegrees (thus $0 < \text{diff}_\tau$), hence we reduced the indegrees of the vertices of at least one block, that is, of at least k vertices. Since $\text{diff}_\tau < k$ it follows that we also reduced the outdegrees of at least one block and thus, such a vertex v exists. If the block of v contains at least $2k$ vertices, then increase the target outdegree of exactly k of these vertices by one. Otherwise, if the block contains less than $2k$ vertices, then increase the target outdegree of all these vertices by one. It follows that $\text{diff}_\tau > k$. Furthermore, observe that $\sum_{v \in V} \tau^+(v) < \sum_{v \in V} \deg_{D+S}^+(v)$, that is, after realizing the target degrees τ , the corresponding k -insertion set S' is still smaller than S .

In the following, we increase the indegrees in two rounds. Observe that if we do not increase outdegrees, then it still holds that $|S'| < |S|$. In the first round, while $\text{diff}_\tau \geq 3k$ do the following:

1. Pick an arbitrary vertex with $\tau^-(v) \leq \Delta_D$.
2. If $|B_\tau(\tau(v))| \leq 2k$, then increase the target indegree $\tau^-(u)$ by one for each $u \in B_\tau(\tau(v))$.
3. Else, it follows that $|B_\tau(\tau(v))| > 2k$. Let $B' \subseteq B_\tau(\tau(v))$ be an arbitrary subset of size exactly k and increase the target indegree $\tau^-(u)$ by one for each $u \in B'$.

Observe that in Point 2 as well as in Point 3 we increase the target indegree of at least k vertices that have the same target degree. Furthermore, in Point 3 we ensure that at least k vertices with the original target degree remain. Hence, the (changed) multiset $\sigma(\tau)$ is still k -anonymous. Furthermore, it is easy to verify that the maximum target indegree is at most $\Delta_D + 1$. Finally, observe that we decrease diff_τ in each iteration by at most $2k$ and, hence, we have $\text{diff}_\tau \geq k$.

In the second round, we have that $k \leq \text{diff}_\tau < 3k$. We simply pick a block $B_\tau(\tau(v))$ with at least $4k$ vertices and increase the target indegree of exactly diff_τ vertices. Since $|V(S)| \geq 4k(\Delta_D + 2)^2$ and there are at most $(\Delta_D + 2)^2$ different degrees in τ (in- and outdegrees between 0 and $\Delta_D + 1$), it follows that there exists such a block of size at least $4k$. Furthermore, observe that after this change in the second round $\sigma(\tau)$ is still k -anonymous and the maximum target indegree is at most $\Delta_D + 2$. Hence, the adjusted target degree function τ fulfills Conditions (I) to (IV) of Lemma 7.

It remains to show the last condition in Lemma 7, that is, Condition (V) stating $s = \sum_{i=1}^n x_i \geq 2\Delta_{D'}^2 + \Delta_{D'}$. Due to the definition of τ (see eq. (3)), it follows that we only

decreased the degrees of vertices with in- or outdegree greater than $\Delta_D + 1$ in $D + S$. Since the target degrees of these vertices is at least $\Delta_D + 1$ (the later changes to τ only increased some degrees), it follows that $V(S)$ is exactly the set of vertices whose target indegree (outdegree) is larger than their indegree (outdegree) in D . Hence,

$$\sum_{v \in V} \tau^+(v) - \deg_D^+(v) \geq |\{v \in V \mid \tau^+(v) > \deg_D^+(v)\}| = |V(S)| \geq 4k(\Delta_D + 2)^2.$$

Since $\Delta_{D'} = \Delta_D + 2$ it follows that Condition (V) is indeed fulfilled. Thus, the set $S' := A'$ realizing τ is a k -insertion set of size less than $|S|$; a contradiction to the fact that S is a minimum size k -insertion set for D . \square \square

Combining Theorems 6, 10 and 24 and Lemma 26, we obtain the following.

Corollary 27. DIGRAPH DEGREE ANONYMITY is fixed-parameter tractable with respect to (k, Δ_D) .

It remains open whether DDA is fixed-parameter tractable with respect to Δ_D . We remark that the problems DDCONC and DDSEQC are both NP-hard for $\Delta_D = 3$. This follows from an adaption of the construction given by Millani [33, Theorem 3.2].

5 Conclusion

We proposed a general framework for digraph degree sequence completion problems and demonstrated its wider applicability in case studies. Somewhat surprisingly, the presumably more technical case of digraphs allowed for some elegant tricks (based on flow computations) that seem not to work for the presumably simpler undirected case. Once having established the framework (see Section 3), the challenges then associated with deriving fixed-parameter tractability and kernelizability results usually boil down to the question for fixed-parameter tractability and (pseudo-)polynomial-time solvability of a simpler problem-specific number problem. While in most cases we could develop polynomial-time algorithms solving these number problems, in the case of DIGRAPH DEGREE ANONYMITY the polynomial-time solvability of the associated number problem remains open. Moreover, a widely open field is to attack weighted versions of our problems. Finally, we believe that due to the fact that many real-world networks are inherently directed (e.g., representing relations such as “follower”, “likes”, or “cites”) further studies (e.g., exploiting special digraph properties) of digraph degree sequence completion problems are desirable.

References

- [1] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009.
- [2] J. Bang-Jensen, A. Frank, and B. Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM Journal on Discrete Mathematics*, 8(2):155–178, 1995.
- [3] J. Bang-Jensen, J. Huang, and X. Zhu. Completing orientations of partially oriented graphs. *CoRR abs/1509.01301*, 2015.

- [4] C. Bazgan, R. Brederbeck, S. Hartung, A. Nichterlein, and G. J. Woeginger. Finding large degree-anonymous subgraphs is hard. *Theoretical Computer Science*, 622:90–110, 2016.
- [5] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra. An algorithm for k -degree anonymity on large networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*, pages 671–675. ACM, 2013.
- [6] W.-K. Chen. On the realization of a (p, s) -digraph with prescribed degrees. *Journal of The Franklin Institute*, 281(5):406–422, 1966.
- [7] S. Chester, B. Kapron, G. Srivastava, and S. Venkatesh. Complexity of social network anonymization. *Social Network Analysis and Mining*, 3(2):151–166, 2013.
- [8] M. Cygan, D. Marx, M. Pilipczuk, M. Pilipczuk, and I. Schlotter. Parameterized complexity of eulerian deletion problems. *Algorithmica*, 68(1):41–61, 2014.
- [9] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
- [10] F. Dorn, H. Moser, R. Niedermeier, and M. Weller. Efficient algorithms for eulerian extension and rural postman. *SIAM Journal on Discrete Mathematics*, 27(1):75–94, 2013.
- [11] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.
- [12] P. Erdős and T. Gallai. Graphs with prescribed degrees of vertices (in Hungarian). *Matematikai Lapok*, 11:264–274, 1960.
- [13] A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
- [14] V. Froese, A. Nichterlein, and R. Niedermeier. Win-win kernelization for degree sequence completion problems. *Journal of Computer and System Sciences*, 82(6):1100–1111, 2016.
- [15] D. Fulkerson. Zero-one matrices with zero trace. *Pacific Journal of Mathematics*, 10(3):831–836, 1960.
- [16] D. Gale. A theorem on flows in networks. *Pacific Journal of Mathematics*, 7:1073–1082, 1957.
- [17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- [18] P. A. Golovach. Editing to a graph of given degrees. *Theoretical Computer Science*, 591:72–84, 2015.
- [19] P. A. Golovach and G. B. Mertzios. Graph editing to a given degree sequence. In *Proceedings of the 11th International Computer Science Symposium in Russia (CSR '16)*, volume 9691 of *LNCS*, pages 177–191. Springer, 2016.
- [20] G. Gutin and A. Yeo. Some parameterized problems on digraphs. *Computer Journal*, 51(3):363–371, 2008.
- [21] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of SIAM*, 10(3):496–506, 1962.
- [22] S. Hartung and A. Nichterlein. NP-hardness and fixed-parameter tractability of realizing degree sequences with directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 29(4):1931–1960, 2015.
- [23] S. Hartung, C. Hoffmann, and A. Nichterlein. Improved upper and lower bound heuristics for degree anonymization in social networks. In *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA '14)*, volume 8504 of *LNCS*, pages 376–387. Springer, 2014.

- [24] S. Hartung, A. Nichterlein, R. Niedermeier, and O. Suchý. A refined complexity analysis of degree anonymization in graphs. *Information and Computation*, 243:249–262, 2015.
- [25] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.
- [26] R. Kannan. Minkowski’s convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.
- [27] D. Kleitman and D. Wang. Algorithms for constructing graphs and digraphs with given valences and factors. *SIAM Journal on Discrete Mathematics*, 6(1):79–88, 1973.
- [28] M. Koseler. Kernelization for degree-constraint editing on directed graphs. Bachelor thesis, TU Berlin, November 2015. URL <http://fpt.akt.tu-berlin.de/publications/theses/BA-marcel-koseler.pdf>.
- [29] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
- [30] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International Conference on World Wide Web (WWW ’08)*, pages 915–924. ACM, 2008.
- [31] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD ’08, pages 93–106. ACM, 2008.
- [32] L. Mathieson and S. Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *Journal of Computer and System Sciences*, 78(1):179–191, 2012.
- [33] M. G. Millani. Algorithms and complexity for degree anonymization in directed graphs. Bachelor thesis, TU Berlin, March 2015. URL <http://fpt.akt.tu-berlin.de/publications/theses/BA-marcelo-millani.pdf>.
- [34] H. Moser and D. M. Thilikos. Parameterized complexity of finding regular induced subgraphs. *Journal of Discrete Algorithms*, 7(2):181–190, 2009.
- [35] J. B. Orlin. Max flows in $o(nm)$ time, or better. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC ’13)*, pages 765–774. ACM, 2013.
- [36] M. Weller, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. On making directed graphs transitive. *Journal of Computer and System Sciences*, 78(2):559–574, 2012.